Direct Digital Synthesis

Introduction to digital Low-Level Radio Frequency Controls in Accelerators

Lab 6

Qiang Du, Larry Doolittle

US Particle Accelerator School
Janurary 23 – 27, 2023

Contents

1 Introduction

In this lab we will learn to design a digital signal synthesizer (DDS) from scratch using the provided verilog module `ph_acc.v`, which is used in production for many LLRF systems as digital local oscillator (LO) and many more cases.

1.1 Glossary

f_MO Master Oscillator frequency, equals to the frequency of operation, or cavity RF frequency.

f_IF Intermediate Frequency. RF signal processing is often performed at a frequency significantly below the frequency of operation. Signal at IF is typically produced by a superheterodyne receiver.

f_LO Local Oscillator frequency.

f_CLK ADC / DAC sampling clock, which is equal to the clock frequency of DSP logic.

DAC Digital-to-analog converter, an electronics device that converts a sequence of digital codes to corresponding analog voltages or currents. We use Analog Devices AD9781 in Zest digitizer board in the LLRF system.

DDS Direct Digital Synthesis. A technique for generating arbitrary frequencies and waveforms from a fixed-frequency clock source.

NCO Numerically-controlled Oscillator

1.2 Binary and non-binary phase accumulator

A DDS in LLRF system is normally built by a CORDIC driven by phase accumulator. Such a phase accumulator is basically a counter with a certain size of step size, so it rolls over at some period, which defines the output frequency f_{DDS}. The linear phase output θ drives a CORDIC for conversion to IQ samples, and a single tone sinusoidal signal is generated with frequency f_{DDS}.

A standard DDS (often referred as NCO) uses a N bit phase accumulator, with a register to define step size (often referred as frequency tunning word, FTW), so the output frequency is:

$$f_{\mathrm{DDS}} = \frac{FTW}{2^N} f_{\mathrm{CLK}}$$

This is restricted to powers of 2 as a denominator because the phase accumulator is a set of bits as wide as the frequency tuning word.

Out `ph_acc.v` has non-binary phase accumulator using the modulo register, so the accumulator is set up to roll over before it reaches full capacity. Every time it rolls over, an extra LSB value is added to the phase accumulator. This is similar to the “Programmable Modulus Mode” in many of the DDS chips such as Analog Devices AD9913, AD9915, and the integrated 12GSPS RF DAC AD9174. The non-binary approach, especially the coarse-binary / fine-with-modulus approach, is mathematically equivalent to Bresenham’s line algorithm.

The 12-bit modulo supports largest known periodicity in a suggested LLRF system, 1427 for JLab. For more normal periodicity, use a multiple to get finer granularity.

For example, in SRRF LLRF, where $N = 20$:

Setting	Relation	Value	Width
IF/Fs	8/11		
`phase_setp_h`	$2^{20} \times 8/11$	762600	20
`phase_step_l`	$(2^{20} \times 8\%11) \times 372$	2976	12
`modulo`	$2^{12} - 372 \times 11$	4	12

In Argonne RIA test:

Setting	Relation	Value	Width
IF/Fs	9/13		
`phase_setp_h`	$2^{20} \times 9/13$	725937	20
`phase_step_l`	$(2^{20} \times 9\%13) \times 315$	945	12
`modulo`	$2^{12} - 315 \times 13$	1	12

If we use the `ph_acc.v` to drive a CORDIC phase, a digital LO in the LLRF system with both $\sin(\theta)$ and $\cos(\theta)$ is then constructed.

2 Exercises

2.1 Calculate registers for desired frequency

The following example provides a way to derive the values for `ph_acc.v` from our class's LLRF LO DDS frequency setting:

Frequency	Derivation	Value	Unit
f_MO		480	MHz
f_IF	f_MO/24	20	MHz
f_LO	F_MO - F_IF	460	MHz
f_CLK	f_LO / 4	115	MHz
f_IF / f_CLK		4 / 23	

We provide two functions for calculating the DDS register values:

```
[1]: from dds import calc_dds, reg2freq
```

To calcuate register values:

[2]:
```
num, den = 4, 23
fclk = 115e6  # Hz

ph, pl, modulo = calc_dds(num, den)
print(f'ph: {ph}, pl: {pl}, modulo: {modulo}')
```

```
ph: 182361, pl: 178, modulo: 2
```

Verify DDS frequency, and print its frequency resolution:

[3]:
```
fdds = reg2freq(ph, pl, modulo, fclk)
print(f'DDS freq: {fdds/1e6:.3f} MHz')
```

```
major resolution:  109.673 Hz
minor resolution:  0.027 Hz
modulo resolution: 0.001 Hz
DDS freq: 20.000 MHz
```

2.2 Simulate DDS verilog module

A basic DDS / NCO module can be as simple as a phase accumulator, with each step size being fractional frequency, so that when it overflows the phase *rotates* a cycle. `ph_acc.v` is designed by Larry Doolittle, with features of 20-bit major resolution and 12-bit minor resolution of frequency, with additional 12-bit modulo register for fractional frequency resolution beyond $2^3 2$ resolution.

Run a behavior simulation and checkout the output using the provided `Makefile`, similar to previous lab exercises. The simulation will generate a data file that records the DDS output `phs_acc.csv` by:

```
$ make ph_acc.csv
```

Or explicitly:

```
$ iverilog -Wall -Wno-timescale -o ph_acc_tb ph_acc_tb.v ph_acc.v
$ vvp -N ph_acc_tb +of=ph_acc.csv
```

Execute the above command, review the source code, testbench and data file.

Plot the time domain waveform and its power spectrum density.

2.3 Derive a different setting for LBNL ALS-U LLRF

Given the ALS-U LLRF setting:

Frequency	Derivation	Value	Unit
f_MO		500	MHz
f_IF	f_MO/12	41.67	MHz

Frequency	Derivation	Value	Unit
f_LO	f_MO - f_IF	458.33	MHz
f_CLK	f_LO / 4	114.58	MHz
f_IF / f_CLK		4 / 11	

Calcuate new values of `ph`, `pl` and `modulo` for digital LO.

2.4 Rerun the DDS verilog simulation

The parameters of `ph`, `pl` and `modulo` can also be passed in the simulation at runtime, by command:

```
$ vvp -N ph_acc_tb +of=ph_acc.csv +ph=<new_ph> +pl=<new_pl> +modulo=<new_modulo>
```

Re-run using your settings and check the new spectrum.

2.5 Build complete DDS with CORDIC

Use the USPAS setting, drive `cordicg_b22.v` which was discussed in CORDIC lab, write a complete DDS module with test bench, run simulation and inspect output spectrum.